Claims

What is claimed is:

1. A computer-automated method for electronic design specification comprising the step of:

specifying at least one resource or functionality using at least one construct in a Resource Description Language (RDL) wherein at least one component or function is specifiable for processing by a high-level synthesis compiler.

2. A computer-automated method for electronic design specification comprising the step of:

specifying an interface of a component in a language-independent manner, wherein a compiler may infer functionality therefrom.

3. A computer-automated method for electronic design specification comprising the step of:

specifying an interface of a component in a language-independent manner, wherein a synthesis compiler automatically processes a component cycle-by-cycle timing behavior without having to specify explicitly a timing diagram for the component.

4. A computer-automated method for electronic design specification comprising the step of:

specifying an interface of a component in a language-independent manner, wherein a synthesis compiler processes a component cycle-by-cycle timing behavior, the component having a fixed latency.

5. A computer-automated method for electronic design specification comprising the step of:

specifying an interface of a component in a language-independent manner, wherein a synthesis compiler processes a component cycle-by-cycle timing behavior, the component having a variable latency.

6. A computer-automated method for electronic design specification comprising the step of:

specifying an interface of a component in a language-independent manner, wherein a synthesis compiler processes a component cycle-by-cycle timing behavior, the component having a pipeline implementation.

7. A computer-automated method for electronic design specification comprising the step of:

specifying an interface of a component in a language-independent manner, wherein a synthesis compiler processes a component cycle-by-cycle timing behavior, the component having a non-pipeline implementation.

8. A computer-automated method for electronic design specification comprising the step of:

specifying an interface of a component in a language-independent manner, wherein a synthesis compiler processes the component cycle-by-cycle timing behavior, the component having a synchronous interface.

9. A computer-automated method for electronic design specification comprising the step of:

specifying an interface of a component in a language-independent manner, wherein a synthesis compiler processes the component cycle-by-cycle timing behavior, the component having an asynchronous interface.

10. A computer-automated method for electronic design specification comprising the step of:

specifying an interface of at least one Intellectual-Property (IP) core in a language-independent manner, wherein a synthesis compiler infers the IP core functionality automatically to instantiate a corresponding IP core.

11. A computer-automated method for electronic design specification comprising the step of:

specifying an interface of a class of Intellectual-Property (IP) cores with one description in a language-independent manner, wherein a synthesis compiler infers an actual IP core to be used automatically and generates a true interface automatically.

12. A computer-automated method for electronic design specification comprising the step of:

specifying a timing interface of at least one Intellectual-Property (IP) core without use of a timing diagram of such IP-core in a language-independent manner, wherein a synthesis compiler infers an IP-core timing behavior for synthesizing an application in which the IP-core is used.

13. A computer-automated method for electronic design specification comprising the step of:

inferring a floorplan of a final hardware on silicon from a generic description stored with one or more Intellectual Property (IP) cores and controlled by a user based on an application for which hardware is being generated.

14. A computer-automated method for electronic design specification comprising the step of:

specifying a function block in a language-independent manner, wherein a synthesis compiler infers a functionality of the function block.

15. A computer-automated method for electronic design specification comprising the step of:

specifying a timing interface of a function block without using a timing diagram of the function block in a language-independent manner, wherein a synthesis compiler infers a timing behavior of the function block for synthesizing an application in which the function block may be used.

16. A computer-automated method for electronic design specification comprising the step of:

specifying a virtual function block comprising one or more function block or Intellectual Property (IP) core, wherein a synthesis compiler infers a functionality of the virtual function block.

17. A computer-automated method for electronic design specification comprising the step of:

specifying a timing interface of a virtual function without using a timing diagram of a virtual function block, wherein a synthesis compiler infers a timing behavior of the virtual function for synthesizing an application in which the virtual function may be used.

18. A computer-automated method for electronic design specification comprising the step of:

specifying a database to store one or more characteristic of an Intellectual Property (IP) core which may be queried by the an RDL specification of the IP core to return information about the IP core.

19. A computer-automated method for electronic design comprising the step of:

generating a hierarchical graph representing one or more required resource requirement of one or more computational block in an application being synthesized.

20. A computer-automated method for electronic design comprising the steps of:

identifying and matching a functionality of a computation block in an application with that of a function block, a virtual function or an Intellectual Property (IP) core to instantiate the computation block.

21. The method of claim 1 wherein:

a resource is used to specify an architecture and a plurality of functionalities.

22. The method of claim 1 wherein:

a unit is used to specify a hardware structure comprising a hierarchical representation of one or more hardware structure.

23. The method of claim 1 wherein:

a UNITDEF value defines or describes a hierarchy of a unit.

24. The method of claim 1 wherein:

a RESOURCEDEF value defines a resource among a set of functionality or associated property.

25. The method of claim 1 wherein:

a RCONNECT value denotes a connection between an origin resource and a destination resource via a connecting resource.

26. The method of claim 1 wherein:

a USES value indicates one or more resource used by a particular resource, the USES value defining at least one virtual resource for building at least one physical resource in an architecture.

27. The method of claim 1 wherein:

a FUNCTIONALITY value specifies a set of one or more basic operator to provide a functionality.

28. The method of claim 1 wherein:

a FUNCTIONALITYDEF value defines a composition of a new functionality.

29. The method of claim 1 wherein:

a DCONNECT value connects a plurality of basic operators while constructing a new functionality.

30. The method of claim 1 wherein:

an INPUT value specifies one or more input node for constructing a new functionality.

31. The method of claim 1 wherein:

an OUTPUT value specifies one or more output node for constructing a new functionality.

32. The method of claim 1 wherein:

an OPT_INPUT value specifies one or more optional input node while constructing a new functionality.

33. The method of claim 1 wherein:

an *if* value specifies an arbitrarily complex connection between a plurality of resources in conjunction with using a *for* value.

34. The method of claim 1 wherein:

a *for* value specifies an arbitrarily complex connection between a plurality of resources in an architecture.

35. The method of claim 1 wherein:

at least one operator in a resource design language (RDL) specifies a hardware and a processing of the hardware.

36. The method of claim 32 wherein:

a hierarchy traversal operator (->) specifies a unit or resource embedded within one or more units by specifying a chain of units hierarchically with the -> operator denoting a child-parent relationship in a hierarchy.

37. The method of claim 32 wherein:

an array operator ([ ]) specifies an array or collection of one or more resource or unit.

38. The method of claim 32 wherein:

a comment operator (//) inserts one or more comment in an architecture file.

39. The method of claim 32 wherein:

operators +,-,* , /,%, ==,!=,>, >=,<,and<= comprise a set of arithmetic or logical operators for constructing one or more expression for use with an if construct selectively to make one or more connection in a for loop.

40. A computer-automated method for electronic design comprising the step of:

generating a hierarchical graph representing a resource target and an interface.

41. A computer-automated method for electronic design comprising the step of:

generating a signature of a functionality graph such that the signature is unique.

42. A computer-automated method for electronic design comprising the step of:

generating a signature of a resource graph such that the signature is unique.

43. A computer-automated method for electronic design comprising the step of:

graph-matching to map a functionality block on an optimally-suited resource graph.

44. A computer-automated method for electronic design comprising the step of:

generating an interface block to enable invocation of a function block for synthesizing an application to a target.

45. A computer-automated method for electronic design specification comprising the steps of:

creating a table structure; and

populating the table structure with one or more function block or Intellectual Property (IP) core, wherein the function block or IP-core area, performance, or interface characteristics comprise one or more capture.

46. A computer-automated method for electronic design comprising the step of:

querying a table structure to obtain an area or performance characteristic of an IP-core or function block optimally to map such area or performance characteristic to one or more resource.

47. The method of claim 43 wherein an area or performance characteristic is useful to optimize a synthesized design.

48. A computer-automated method for electronic design specification comprising the steps of:

specifying and querying an interface or characteristic of an Intellectual Property (IP) core for a Field Programmable Gate Array (FPGA) generator according substantially to a Xilinx vendor format and specification to incorporate in a synthesized design on a target.

49. A computer-automated method for electronic design specification comprising the step of:

specifying and query an interface and characteristic of an Intellectual Property (IP) core for a digital signal processor (DSP) builder according substantially to an Altera vendor format and specification to incorporate in a synthesized design on a target.

50. A computer-automated method for electronic design specification comprising the step of:

specifying and querying an interface and characteristic of an Intellectual Property (IP) core for a tool according substantially to a Quicklogic format and specification to incorporate in a synthesized design on a target.

51. A computer-automated method for electronic design specification comprising the step of:

specifying and querying an interface and the characteristics of an Intellectual Property (IP) core from a soft core vendor for use in an Application Specific Integrated Circuit (ASIC) or Field Programmable Gate Array (FPGA) architecture.

52. A computer-automated method to synthesize an application to a target comprising the steps of:

reading and parsing a resource description language (RDL) description of a target;

generating an RDL Abstract Syntax Tree;

generating a resource graph for the target;

generating a functionality graph from an application;

generating a signature of one or more node of the functionality graph;

querying an architecture interface of the resource graph;

generating one or more signature of a functionality of the node in the resource graph;

optimally matching one or more signature of the node of the functionality and resource graphs to map optimally a function to a resource of the target; and

generating an interface in a synthesized application to invoke one or more function block, Intellectual Property core (IP-core), or component on the target.